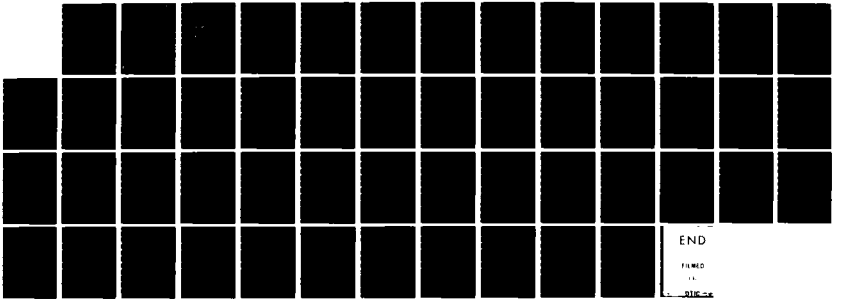
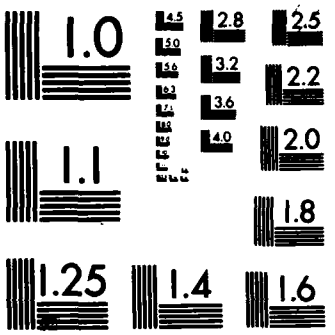
AD-A130 720 A FULLY AUTOMATIC PROOF OF AN ORDERED TREE INSERTION FUNCTION(U) TEXAS UNIV AT AUSTIN INST FOR COMPUTING SCIENCE AND COMPUTER A.. R S BOYER ET AL. MAY 83 1/1

UNCLASSIFIED ICSCA-CMP-36 N00014-81-K-0634 F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(12)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: ICSCA-CMP-36 ✓ 2. GOVT ACCESSION NO.: AD-A130720	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): A FULLY AUTOMATIC PROOF OF AN ORDERED TREE INSERTION FUNCTION	5. TYPE OF REPORT & PERIOD COVERED: Technical
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Robert S. Boyer & J Strother Moore	8. CONTRACT OR GRANT NUMBER(s): MCS-8202943; N00014-81-K-0634
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Institute for Computing Science & Computer Applications, The University of Texas at Austin, Main Building 2100, Austin, Texas 78712	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR 049-500
11. CONTROLLING OFFICE NAME AND ADDRESS: Software Systems Science, National Science Found., Washington, DC 20550; Office of Naval Research, 800 N. Quincy St, Arlington, VA 22217	12. REPORT DATE: May, 1983 13. NUMBER OF PAGES: 50 pages
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The authors define the concept of an ordered binary tree, the concept that a number occurs in such a tree, and a function that is purported to insert a number into such a tree. They then show a fully automatic proof that the insertion function is correct. The machine proves the following three theorems without any help whatsoever. First, the recursive equation describing the insertion function is satisfied by one and only one function - in particular, the machine proves that the recursion always terminates. Second, if X is an ordered tree and L is a number, then the result of inserting L into X is an ordered tree. Third, if X is an ordered tree (cont.)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (continued)

and L and K are numbers, then L occurs in the result of inserting K into X if and only if either L=K or L already occurred in X.

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Table of Contents

Abstract

The authors

~~We~~ define the concept of an ordered binary tree, the concept that a number occurs in such a tree, and a function that is purported to insert a number into such a tree. ~~We~~ then show a fully automatic proof that the insertion function is correct. The machine proves the following three theorems without any help whatsoever. First, the recursive equation describing the insertion function is satisfied by one and only one function -- in particular, the machine proves that the recursion always terminates. Second, if X is an ordered tree and L is a number, then the result of inserting L into X is an ordered tree. Third, if X is an ordered tree and L and K are numbers, then L occurs in the result of inserting K into X if and only if either L=K or L already occurred in X.

1. Introduction

In a recent technical report [7] Eriksson, Johansson, and Tarnlund state that "no automatic theorem prover is probably able to prove" the correctness of a particular ordered tree insertion algorithm. They describe a proof-checking system and mention a proof within the system of the theorem with more than six hundred steps. They mention that the possiblity of such proofs "would be another advantage of predicate logic as a formalism for programming."

In this paper, we define the insertion algorithm as a recursive function and demonstrate that the current version of the Boyer-Moore theorem-prover, described in [1, 2], can prove its correctness completely automatically.

In the next section we list the commands typed to the theorem-prover to cause it to prove the required theorems. In the section after that we include the theorem-prover's output in response to each command. We then close with some pertinent remarks.

2. Input to the Theorem-Prover

The theorem-prover was started up in its most elementary state, in which it knew the axioms of our theory as described in Chapter III of [1]. We then typed six commands.

The first introduces a new "shell" or abstract data type as described in [1]. The "shell" principle is a schematic way to describe axioms for an inductively constructed data type. Roughly speaking the axioms define a new type of object called an ordered triple. They provide an empty ordered triple, constructed by the constant function ET of no arguments, and nonempty ordered triples constructed by the function OT of three arguments. The new predicate OTP is axiomatized to return T or F according to whether its argument is an (empty or nonempty) ordered triple. The constructor OT function "expects" its arguments to be, respectively, an ordered triple, a natural number, and an ordered triple. If the first or third argument is not an ordered triple, it is "defaulted" to the empty triple. If the second argument is not a number it is "defaulted" to 0. The new functions LT, LABEL, and RT, when applied to the result of (OT x y z), return x, y, and z respectively (or the appropriate defaults). Syntactic restrictions on the shell principle insure that the resulting set of axioms consistently extends the existing theory.

The next three commands recursively define the ORDERED.TREE predicate, the INSERT function, and the OCCUR.TREE predicate. These definitions, proposed as new definitional axioms in the theory, should be self-explanatory to anyone familiar with LISP. The theorem-prover must show that the recursion in these definitions is well-founded in order to admit these equations as definitional axioms. As proved in [1], if so admitted there exists one and only one function satisfying each equation.

Finally, we command the theorem-prover to prove the theorems that (a) INSERT produces an ORDERED.TREE when called on one, and (b) the members of the new tree are precisely the members of the old one plus the element inserted.

Here are the commands themselves. This constitutes all of the user supplied information.

```
Shell Definition.
Add the shell OT of three arguments with
bottom object ET,
recognizer OTP,
accessors LT, LABEL, and RT,
type restrictions (ONE.OF OTP), (ONE.OF NUMBERP), and (ONE.OF OTP),
and default values ET, ZERO, and ET.

Definition.
(ORDERED.TREE X)
   =
(IF
 (NOT (OTP X))
 F
 (IF (EQUAL (ET) X)
     T
     (AND (ORDERED.TREE (LT X))
          (ORDERED.TREE (RT X))
          (OR (EQUAL (LT X) (ET))
              (LESSP (LABEL (LT X)) (LABEL X)))
          (OR (EQUAL (RT X) (ET))
              (LESSP (LABEL X) (LABEL (RT X)))))))

Definition.
(INSERT L X)
   =
(IF (NOT (OTP X))
    (ET)
    (IF (EQUAL X (ET))
        (OT (ET) L (ET))
        (IF (LESSP (LABEL X) L)
            (OT (LT X)
                (LABEL X)
                (INSERT L (RT X)))
            (IF (LESSP L (LABEL X))
                (OT (INSERT L (LT X))
                    (LABEL X)
                    (RT X))
                X))))

Definition.
(OCCUR.TREE L X)
   =
(IF (NOT (OTP X))
    F
    (IF (EQUAL X (ET))
        F
        (IF (EQUAL L (LABEL X))
            T
            (OR (OCCUR.TREE L (LT X))
                (OCCUR.TREE L (RT X))))))

Theorem.  ORDERED.TREE.INSERT:
(IMPLIES (AND (NUMBERP L) (ORDERED.TREE X))
         (ORDERED.TREE (INSERT L X)))
```

```
Theorem.  OCCUR.TREE.INSERT:
(IMPLIES
        (AND (NUMBERP L)
             (NUMBERP K)
             (ORDERED.TREE X))
        (EQUAL (OCCUR.TREE L (INSERT K X))
               (OR (EQUAL L K) (OCCUR.TREE L X))))
```

3. The Output by the Theorem-Prover

We here give the theorem-prover's response to each command. For the reader's convenience we repeat each command. The lines of the form

[n cns / x s + y gc + z io (= m @ k)]

give performance statistics. The interpretation of the numbers are:

n = number of conses used by INTERLISP

x = number of DEC 2060 cpu seconds spent inside the theorem-prover

y = number of DEC 2060 cpu second in garbage collector

z = number of DEC 2060 cpu seconds in output routines

m = total elapsed time in seconds

k = load average

Except for the commands repeated below, all of the text in the following subsections was generated by the theorem-prover. The proofs appear long because there are many cases to consider and the theorem-prover verbosely prints each of the cases. The number of pages of output is not a good measure of the difficulty of the proofs. A better measure of the difficulty of the proofs is the amount of user-typein necessary to lead the machine to the proofs. Except for the commands defining the concepts necessary to state the theorems, the user contributed nothing to these proofs.

3.1. The Add Shell Command

```
Shell Definition.
Add the shell OT of three arguments with
bottom object ET,
recognizer OTP,
accessors LT, LABEL, and RT,
type restrictions (ONE.OF OTP), (ONE.OF NUMBERP), and (ONE.OF OTP),
and default values ET, ZERO, and ET.


[17911 cns / 25.4 s + 0.0 gc + 0.0 io (= 33 @ 2)]
OT
```

3.2. ORDERED.TREE

```
Definition.
(ORDERED.TREE X)
   =
(IF
 (NOT (OTP X))
 F
 (IF (EQUAL (ET) X)
     T
     (AND (ORDERED.TREE (LT X))
          (ORDERED.TREE (RT X))
          (OR (EQUAL (LT X) (ET))
              (LESSP (LABEL (LT X)) (LABEL X)))
          (OR (EQUAL (RT X) (ET))
              (LESSP (LABEL X) (LABEL (RT X)))))))
```

Linear arithmetic and the lemmas RT.LESSP and LT.LESSP inform us that the measure (COUNT X) decreases according to the well-founded relation LESSP in each recursive call. Hence, ORDERED.TREE is accepted under the principle of definition. Observe that:

```
(OR (FALSEP (ORDERED.TREE X))
    (TRUEP (ORDERED.TREE X)))
```

is a theorem.

```
[4467 cns / 5.6 s + 0.0 gc + .4 io (= 22 @ 2)]
ORDERED.TREE
```

3.3. INSERT

```
Definition.
(INSERT L X)
   =
(IF (NOT (OTP X))
    (ET)
    (IF (EQUAL X (ET))
        (OT (ET) L (ET))
        (IF (LESSP (LABEL X) L)
            (OT (LT X)
                (LABEL X)
                (INSERT L (RT X)))
            (IF (LESSP L (LABEL X))
                (OT (INSERT L (LT X))
                    (LABEL X)
                    (RT X))
                X))))
```

Linear arithmetic and the lemmas RT.LESSP and LT.LESSP establish that the measure (COUNT X) decreases according to the well-founded relation LESSP in each recursive call. Hence, INSERT is accepted under the principle of definition. Observe that (OTP (INSERT L X)) is a theorem.

```
[3397 cns / 2.6 s + 0.0 gc + .3 io (= 5 @ 2)]
INSERT
```

3.4. OCCUR.TREE

```
Definition.
(OCCUR.TREE L X)
   =
(IF (NOT (OTP X))
    F
    (IF (EQUAL X (ET))
        F
        (IF (EQUAL L (LABEL X))
            T
            (OR (OCCUR.TREE L (LT X))
                (OCCUR.TREE L (RT X))))))
```

Linear arithmetic and the lemmas RT.LESSP and LT.LESSP establish that the measure (COUNT X) decreases according to the well-founded relation LESSP in each recursive call. Hence, OCCUR.TREE is accepted under the definitional principle. Observe that:

```
(OR (FALSEP (OCCUR.TREE L X))
    (TRUEP (OCCUR.TREE L X)))
```

is a theorem.

```
[2106 cns / 2.0 s + 0.0 gc + .4 io (= 5 @ 2)]
OCCUR.TREE
```

3.5. ORDERED.TREE.INSERT

```
Theorem.  ORDERED.TREE.INSERT:
(IMPLIES (AND (NUMBERP L) (ORDERED.TREE X))
         (ORDERED.TREE (INSERT L X)))
```

Name the conjecture *1.

Let us appeal to the induction principle. Two inductions are suggested by terms in the conjecture. However, they merge into one likely candidate induction. We will induct according to the following scheme:

```
(AND (IMPLIES (NOT (OTP X)) (p L X))
     (IMPLIES (AND (OTP X) (EQUAL (ET) X))
              (p L X))
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (p L (LT X))
                   (p L (RT X)))
              (p L X))).
```

Linear arithmetic and the lemmas RT.LESSP and LT.LESSP inform us that the measure (COUNT X) decreases according to the well-founded relation LESSP in each induction step of the scheme. The above induction scheme generates six new formulas:

```
Case 6. (IMPLIES (AND (NOT (OTP X))
                      (NUMBERP L)
                      (ORDERED.TREE X))
                 (ORDERED.TREE (INSERT L X))).
```

This simplifies, opening up the definition of ORDERED.TREE, to:

T.

```
Case 5. (IMPLIES (AND (OTP X)
                      (EQUAL (ET) X)
                      (NUMBERP L)
                      (ORDERED.TREE X))
                 (ORDERED.TREE (INSERT L X))),
```

which simplifies, applying the lemmas RT.OT and LT.OT, and expanding the functions OTP, ORDERED.TREE, EQUAL, and INSERT, to:

```
      T.
```

```
Case 4. (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (ORDERED.TREE (LT X)))
                      (NOT (ORDERED.TREE (RT X)))
                      (NUMBERP L)
                      (ORDERED.TREE X))
                 (ORDERED.TREE (INSERT L X))),
```

which simplifies, expanding ORDERED.TREE, to:

```
      T.
```

```
Case 3. (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (ORDERED.TREE (INSERT L (LT X)))
                      (NOT (ORDERED.TREE (RT X)))
                      (NUMBERP L)
                      (ORDERED.TREE X))
                 (ORDERED.TREE (INSERT L X))),
```

which we simplify, unfolding ORDERED.TREE, to:

```
      T.
```

```
Case 2. (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (ORDERED.TREE (LT X)))
                      (ORDERED.TREE (INSERT L (RT X)))
                      (NUMBERP L)
                      (ORDERED.TREE X))
                 (ORDERED.TREE (INSERT L X))),
```

which we simplify, unfolding the definition of ORDERED.TREE, to:

```
      T.
```

```
Case 1. (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (ORDERED.TREE (INSERT L (LT X)))
                      (ORDERED.TREE (INSERT L (RT X)))
                      (NUMBERP L)
                      (ORDERED.TREE X))
                 (ORDERED.TREE (INSERT L X))).
```

This simplifies, opening up ORDERED.TREE and INSERT, to 12 new formulas:

```
Case 1.12.
        (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (ORDERED.TREE (INSERT L (LT X)))
                      (ORDERED.TREE (INSERT L (RT X)))
                      (NUMBERP L)
```

```
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (EQUAL (RT X) (ET))
                    (LEQ L (LABEL X))
                    (LESSP L (LABEL X)))
               (ORDERED.TREE (OT (INSERT L (LT X))
                                 (LABEL X)
                                 (ET)))).
```

This again simplifies, applying the lemmas RT.OT, LT.OT, and LABEL.OT, and opening up the definitions of EQUAL, OTP, INSERT, and ORDERED.TREE, to:

```
     T.
```

```
Case 1.11.
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (ORDERED.TREE (INSERT L (LT X)))
                   (ORDERED.TREE (INSERT L (RT X)))
                   (NUMBERP L)
                   (ORDERED.TREE (LT X))
                   (ORDERED.TREE (RT X))
                   (EQUAL (LT X) (ET))
                   (EQUAL (RT X) (ET))
                   (LEQ L (LABEL X))
                   (LEQ (LABEL X) L))
              (ORDERED.TREE X)).
```

This again simplifies, using linear arithmetic, to:

```
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (ORDERED.TREE (INSERT (LABEL X) (ET)))
                   (ORDERED.TREE (INSERT (LABEL X) (ET)))
                   (NUMBERP (LABEL X))
                   (ORDERED.TREE (ET))
                   (ORDERED.TREE (ET))
                   (EQUAL (LT X) (ET))
                   (EQUAL (RT X) (ET))
                   (LEQ (LABEL X) (LABEL X))
                   (LEQ (LABEL X) (LABEL X)))
              (ORDERED.TREE X)),
```

which again simplifies, applying RT.OT and LT.OT, and unfolding EQUAL, OTP, INSERT, and ORDERED.TREE, to two new conjectures:

```
Case 1.11.2.
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (EQUAL (LT X) (ET))
                   (EQUAL (RT X) (ET))
                   (LEQ (LABEL X) (LABEL X)))
              (ORDERED.TREE (LT X))).
```

But this again simplifies, expanding the definition of ORDERED.TREE, to:

```
     T.
```

```
Case 1.11.1.
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (EQUAL (LT X) (ET))
```

```
                    (EQUAL (RT X) (ET))
                    (LEQ (LABEL X) (LABEL X)))
               (ORDERED.TREE (RT X))).
```

This simplifies again, expanding the definition of ORDERED.TREE, to:

T.

Case 1.10.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (ORDERED.TREE (INSERT L (LT X)))
              (ORDERED.TREE (INSERT L (RT X)))
              (NUMBERP L)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (EQUAL (RT X) (ET))
              (LESSP (LABEL X) L))
         (ORDERED.TREE (OT (ET)
                           (LABEL X)
                           (INSERT L (RT X))))),
```

which again simplifies, applying RT.OT, LT.OT, and LABEL.OT, and expanding EQUAL, OTP, INSERT, and ORDERED.TREE, to:

T.

Case 1.9.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (ORDERED.TREE (INSERT L (LT X)))
              (ORDERED.TREE (INSERT L (RT X)))
              (NUMBERP L)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (EQUAL (RT X) (ET))
              (LEQ L (LABEL X))
              (LESSP L (LABEL X)))
         (ORDERED.TREE (OT (INSERT L (LT X))
                           (LABEL X)
                           (ET)))).
```

This simplifies again, applying RT.OT, LT.OT, and LABEL.OT, and expanding the definitions of EQUAL, OTP, INSERT, and ORDERED.TREE, to:

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (ORDERED.TREE (INSERT L (LT X)))
              (NUMBERP L)
              (ORDERED.TREE (LT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (EQUAL (RT X) (ET))
              (LEQ L (LABEL X))
              (LESSP L (LABEL X))
              (NOT (EQUAL (INSERT L (LT X)) (ET))))
         (LESSP (LABEL (INSERT L (LT X)))
                (LABEL X))).
```

Applying the lemma LT/LABEL/RT.ELIM, replace X by (OT Z W V) to eliminate (LT X), (RT X), and (LABEL X) and Z by (OT X1 D C) to eliminate (LABEL Z), (RT Z), and (LT Z). We employ the type restriction lemma noted when LABEL

was introduced, the type restriction lemma noted when RT was introduced, and the type restriction lemma noted when LT was introduced to constrain the new variables. This generates three new conjectures:

```
Case 1.9.3.
        (IMPLIES (AND (EQUAL X (ET))
                      (OTP X)
                      (NOT (EQUAL (ET) X))
                      (ORDERED.TREE (INSERT L (LT X)))
                      (NUMBERP L)
                      (ORDERED.TREE (LT X))
                      (LESSP (LABEL (LT X)) (LABEL X))
                      (EQUAL (RT X) (ET))
                      (LEQ L (LABEL X))
                      (LESSP L (LABEL X))
                      (NOT (EQUAL (INSERT L (LT X)) (ET))))
                 (LESSP (LABEL (INSERT L (LT X)))
                        (LABEL X))).
```

Of course, this simplifies further, clearly, to:

```
        T.
```

```
Case 1.9.2.
        (IMPLIES (AND (EQUAL Z (ET))
                      (OTP Z)
                      (OTP V)
                      (NUMBERP W)
                      (NOT (EQUAL (OT Z W V) (ET)))
                      (NOT (EQUAL (ET) (OT Z W V)))
                      (ORDERED.TREE (INSERT L Z))
                      (NUMBERP L)
                      (ORDERED.TREE Z)
                      (LESSP (LABEL Z) W)
                      (EQUAL V (ET))
                      (LEQ L W)
                      (LESSP L W)
                      (NOT (EQUAL (INSERT L Z) (ET))))
                 (LESSP (LABEL (INSERT L Z)) W)).
```

But this further simplifies, applying RT.OT, LT.OT, and LABEL.OT, and expanding OTP, EQUAL, INSERT, ORDERED.TREE, LABEL, and LESSP, to:

```
        T.
```

```
Case 1.9.1.
        (IMPLIES (AND (NUMBERP D)
                      (OTP C)
                      (OTP X1)
                      (NOT (EQUAL (OT X1 D C) (ET)))
                      (OTP V)
                      (NUMBERP W)
                      (NOT (EQUAL (OT (OT X1 D C) W V) (ET)))
                      (NOT (EQUAL (ET) (OT (OT X1 D C) W V)))
                      (ORDERED.TREE (INSERT L (OT X1 D C)))
                      (NUMBERP L)
                      (ORDERED.TREE (OT X1 D C))
                      (LESSP D W)
                      (EQUAL V (ET))
                      (LEQ L W)
                      (LESSP L W)
                      (NOT (EQUAL (INSERT L (OT X1 D C)) (ET))))
                 (LESSP (LABEL (INSERT L (OT X1 D C)))
```

```
                    W)),
```

which we further simplify, applying RT.OT, LT.OT, and LABEL.OT, and expanding the functions OTP, INSERT, ORDERED.TREE, and EQUAL, to:

```
     T.

Case 1.8.
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (ORDERED.TREE (INSERT L (LT X)))
                   (ORDERED.TREE (INSERT L (RT X)))
                   (NUMBERP L)
                   (ORDERED.TREE (LT X))
                   (ORDERED.TREE (RT X))
                   (LESSP (LABEL (LT X)) (LABEL X))
                   (EQUAL (RT X) (ET))
                   (LEQ L (LABEL X))
                   (LEQ (LABEL X) L))
              (ORDERED.TREE X)),
```

which again simplifies, using linear arithmetic, to the conjecture:

```
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (ORDERED.TREE (INSERT (LABEL X) (LT X)))
                   (ORDERED.TREE (INSERT (LABEL X) (ET)))
                   (NUMBERP (LABEL X))
                   (ORDERED.TREE (LT X))
                   (ORDERED.TREE (ET))
                   (LESSP (LABEL (LT X)) (LABEL X))
                   (EQUAL (RT X) (ET))
                   (LEQ (LABEL X) (LABEL X))
                   (LEQ (LABEL X) (LABEL X)))
              (ORDERED.TREE X)).
```

This simplifies again, applying RT.OT and LT.OT, and expanding EQUAL, OTP, INSERT, and ORDERED.TREE, to the new conjecture:

```
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (ORDERED.TREE (INSERT (LABEL X) (LT X)))
                   (ORDERED.TREE (LT X))
                   (LESSP (LABEL (LT X)) (LABEL X))
                   (EQUAL (RT X) (ET))
                   (LEQ (LABEL X) (LABEL X)))
              (ORDERED.TREE (RT X))).
```

This simplifies again, expanding ORDERED.TREE, to:

```
     T.

Case 1.7.
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (ORDERED.TREE (INSERT L (LT X)))
                   (ORDERED.TREE (INSERT L (RT X)))
                   (NUMBERP L)
                   (ORDERED.TREE (LT X))
                   (ORDERED.TREE (RT X))
                   (LESSP (LABEL (LT X)) (LABEL X))
                   (EQUAL (RT X) (ET))
                   (LESSP (LABEL X) L))
```

```
                (ORDERED.TREE (OT (LT X)
                                  (LABEL X)
                                  (INSERT L (RT X))))),
```

which again simplifies, applying RT.OT, LT.OT, and LABEL.OT, and unfolding the definitions of EQUAL, OTP, INSERT, and ORDERED.TREE, to:

```
      T.
```

```
Case 1.6.
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (ORDERED.TREE (INSERT L (LT X)))
                    (ORDERED.TREE (INSERT L (RT X)))
                    (NUMBERP L)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (LEQ L (LABEL X))
                    (LESSP L (LABEL X)))
               (ORDERED.TREE (OT (INSERT L (LT X))
                                 (LABEL X)
                                 (RT X)))).
```

This again simplifies, applying RT.OT, LT.OT, and LABEL.OT, and expanding the functions EQUAL, OTP, INSERT, and ORDERED.TREE, to:

```
      T.
```

```
Case 1.5.
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (ORDERED.TREE (INSERT L (LT X)))
                    (ORDERED.TREE (INSERT L (RT X)))
                    (NUMBERP L)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (LEQ L (LABEL X))
                    (LEQ (LABEL X) L))
               (ORDERED.TREE X)).
```

This simplifies again, using linear arithmetic, to:

```
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (ORDERED.TREE (INSERT (LABEL X) (ET)))
                    (ORDERED.TREE (INSERT (LABEL X) (RT X)))
                    (NUMBERP (LABEL X))
                    (ORDERED.TREE (ET))
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (LEQ (LABEL X) (LABEL X))
                    (LEQ (LABEL X) (LABEL X)))
               (ORDERED.TREE X)),
```

which we again simplify, applying the lemmas RT.OT and LT.OT, and unfolding EQUAL, OTP, INSERT, and ORDERED.TREE, to the conjecture:

```
      (IMPLIES (AND (OTP X)
```

```
                    (NOT (EQUAL (ET) X))
                    (ORDERED.TREE (INSERT (LABEL X) (RT X)))
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (LEQ (LABEL X) (LABEL X)))
               (ORDERED.TREE (LT X))),
```

which we again simplify, expanding the function ORDERED.TREE, to:

```
     T.
```

```
Case 1.4.
       (IMPLIES (AND (OTP X)
                     (NOT (EQUAL (ET) X))
                     (ORDERED.TREE (INSERT L (LT X)))
                     (ORDERED.TREE (INSERT L (RT X)))
                     (NUMBERP L)
                     (ORDERED.TREE (LT X))
                     (ORDERED.TREE (RT X))
                     (EQUAL (LT X) (ET))
                     (LESSP (LABEL X) (LABEL (RT X)))
                     (LESSP (LABEL X) L))
                (ORDERED.TREE (OT (ET)
                                  (LABEL X)
                                  (INSERT L (RT X))))),
```

which again simplifies, applying RT.OT, LT.OT, and LABEL.OT, and expanding EQUAL, OTP, INSERT, and ORDERED.TREE, to:

```
       (IMPLIES (AND (OTP X)
                     (NOT (EQUAL (ET) X))
                     (ORDERED.TREE (INSERT L (RT X)))
                     (NUMBERP L)
                     (ORDERED.TREE (RT X))
                     (EQUAL (LT X) (ET))
                     (LESSP (LABEL X) (LABEL (RT X)))
                     (LESSP (LABEL X) L)
                     (NOT (EQUAL (INSERT L (RT X)) (ET))))
                (LESSP (LABEL X)
                       (LABEL (INSERT L (RT X))))).
```

Applying the lemma LT/LABEL/RT.ELIM, replace X by (OT W V Z) to eliminate (RT X), (LABEL X), and (LT X) and Z by (OT X1 D C) to eliminate (LABEL Z), (RT Z), and (LT Z). We employ the type restriction lemma noted when LABEL was introduced, the type restriction lemma noted when RT was introduced, and the type restriction lemma noted when LT was introduced to restrict the new variables. We would thus like to prove three new goals:

```
Case 1.4.3.
        (IMPLIES (AND (EQUAL X (ET))
                      (OTP X)
                      (NOT (EQUAL (ET) X))
                      (ORDERED.TREE (INSERT L (RT X)))
                      (NUMBERP L)
                      (ORDERED.TREE (RT X))
                      (EQUAL (LT X) (ET))
                      (LESSP (LABEL X) (LABEL (RT X)))
                      (LESSP (LABEL X) L)
                      (NOT (EQUAL (INSERT L (RT X)) (ET))))
                 (LESSP (LABEL X)
                        (LABEL (INSERT L (RT X))))).
```

This further simplifies, clearly, to:

T.

Case 1.4.2.

```
(IMPLIES (AND (EQUAL Z (ET))
              (OTP Z)
              (NUMBERP V)
              (OTP W)
              (NOT (EQUAL (OT W V Z) (ET)))
              (NOT (EQUAL (ET) (OT W V Z)))
              (ORDERED.TREE (INSERT L Z))
              (NUMBERP L)
              (ORDERED.TREE Z)
              (EQUAL W (ET))
              (LESSP V (LABEL Z))
              (LESSP V L)
              (NOT (EQUAL (INSERT L Z) (ET))))
         (LESSP V (LABEL (INSERT L Z)))).
```

This further simplifies, applying RT.OT and LT.OT, and unfolding the definitions of OTP, EQUAL, INSERT, ORDERED.TREE, LABEL, and LESSP, to:

T.

Case 1.4.1.

```
(IMPLIES (AND (NUMBERP D)
              (OTP C)
              (OTP X1)
              (NOT (EQUAL (OT X1 D C) (ET)))
              (NUMBERP V)
              (OTP W)
              (NOT (EQUAL (OT W V (OT X1 D C)) (ET)))
              (NOT (EQUAL (ET) (OT W V (OT X1 D C))))
              (ORDERED.TREE (INSERT L (OT X1 D C)))
              (NUMBERP L)
              (ORDERED.TREE (OT X1 D C))
              (EQUAL W (ET))
              (LESSP V D)
              (LESSP V L)
              (NOT (EQUAL (INSERT L (OT X1 D C)) (ET))))
         (LESSP V
                (LABEL (INSERT L (OT X1 D C))))).
```

However this further simplifies, appealing to the lemmas RT.OT, LT.OT, and LABEL.OT, and opening up OTP, INSERT, ORDERED.TREE, and EQUAL, to:

T.

Case 1.3.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (ORDERED.TREE (INSERT L (LT X)))
              (ORDERED.TREE (INSERT L (RT X)))
              (NUMBERP L)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (LESSP (LABEL X) (LABEL (RT X)))
              (LEQ L (LABEL X))
              (LESSP L (LABEL X)))
         (ORDERED.TREE (OT (INSERT L (LT X))
                           (LABEL X)
```

```
                                    (RT X))))).
```

This again simplifies, applying LABEL.OT, RT.OT, and LT.OT, and opening up the definition of ORDERED.TREE, to the formula:

```
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (ORDERED.TREE (INSERT L (LT X)))
                    (ORDERED.TREE (INSERT L (RT X)))
                    (NUMBERP L)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (LESSP (LABEL (LT X)) (LABEL X))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (LEQ L (LABEL X))
                    (LESSP L (LABEL X))
                    (NOT (EQUAL (INSERT L (LT X)) (ET))))
               (LESSP (LABEL (INSERT L (LT X)))
                      (LABEL X))).
```

Applying the lemma LT/LABEL/RT.ELIM, we now replace X by (OT Z W V) to eliminate (LT X), (RT X), and (LABEL X), Z by (OT X1 D C) to eliminate (LABEL Z), (RT Z), and (LT Z), V by (OT V1 Z Z1) to eliminate (LABEL V), (RT V), and (LT V), and V by (OT X1 D C) to eliminate (LABEL V), (RT V), and (LT V). We use the type restriction lemma noted when LABEL was introduced, the type restriction lemma noted when RT was introduced, and the type restriction lemma noted when LT was introduced to restrict the new variables. This produces five new goals:

```
Case 1.3.5.
       (IMPLIES (AND (EQUAL X (ET))
                     (OTP X)
                     (NOT (EQUAL (ET) X))
                     (ORDERED.TREE (INSERT L (LT X)))
                     (ORDERED.TREE (INSERT L (RT X)))
                     (NUMBERP L)
                     (ORDERED.TREE (LT X))
                     (ORDERED.TREE (RT X))
                     (LESSP (LABEL (LT X)) (LABEL X))
                     (LESSP (LABEL X) (LABEL (RT X)))
                     (LEQ L (LABEL X))
                     (LESSP L (LABEL X))
                     (NOT (EQUAL (INSERT L (LT X)) (ET))))
                (LESSP (LABEL (INSERT L (LT X)))
                       (LABEL X))),
```

which we further simplify, clearly, to:

```
       T.

Case 1.3.4.
       (IMPLIES (AND (EQUAL V (ET))
                     (EQUAL Z (ET))
                     (OTP Z)
                     (OTP V)
                     (NUMBERP W)
                     (NOT (EQUAL (OT Z W V) (ET)))
                     (NOT (EQUAL (ET) (OT Z W V)))
                     (ORDERED.TREE (INSERT L Z))
                     (ORDERED.TREE (INSERT L V))
                     (NUMBERP L)
                     (ORDERED.TREE Z)
                     (ORDERED.TREE V)
```

```
                    (LESSP (LABEL Z) W)
                    (LESSP W (LABEL V))
                    (LEQ L W)
                    (LESSP L W)
                    (NOT (EQUAL (INSERT L Z) (ET))))
               (LESSP (LABEL (INSERT L Z)) W)).
```

However this simplifies further, using linear arithmetic, to:

T.

Case 1.3.3.

```
(IMPLIES (AND (NUMBERP D)
              (OTP C)
              (OTP X1)
              (NOT (EQUAL (OT X1 D C) (ET)))
              (EQUAL Z (ET))
              (OTP Z)
              (NUMBERP W)
              (NOT (EQUAL (OT Z W (OT X1 D C)) (ET)))
              (NOT (EQUAL (ET) (OT Z W (OT X1 D C))))
              (ORDERED.TREE (INSERT L Z))
              (ORDERED.TREE (INSERT L (OT X1 D C)))
              (NUMBERP L)
              (ORDERED.TREE Z)
              (ORDERED.TREE (OT X1 D C))
              (LESSP (LABEL Z) W)
              (LESSP W D)
              (LEQ L W)
              (LESSP L W)
              (NOT (EQUAL (INSERT L Z) (ET))))
         (LESSP (LABEL (INSERT L Z)) W)),
```

which we further simplify, applying RT.OT, LT.OT, and LABEL.OT, and unfolding the definitions of OTP, EQUAL, INSERT, ORDERED.TREE, LABEL, and LESSP, to:

T.

Case 1.3.2.

```
(IMPLIES (AND (EQUAL V (ET))
              (NUMBERP D)
              (OTP C)
              (OTP X1)
              (NOT (EQUAL (OT X1 D C) (ET)))
              (OTP V)
              (NUMBERP W)
              (NOT (EQUAL (OT (OT X1 D C) W V) (ET)))
              (NOT (EQUAL (ET) (OT (OT X1 D C) W V)))
              (ORDERED.TREE (INSERT L (OT X1 D C)))
              (ORDERED.TREE (INSERT L V))
              (NUMBERP L)
              (ORDERED.TREE (OT X1 D C))
              (ORDERED.TREE V)
              (LESSP D W)
              (LESSP W (LABEL V))
              (LEQ L W)
              (LESSP L W)
              (NOT (EQUAL (INSERT L (OT X1 D C)) (ET))))
         (LESSP (LABEL (INSERT L (OT X1 D C)))
                W)).
```

But this further simplifies, applying RT.OT, LT.OT, and LABEL.OT, and

opening up OTP, INSERT, EQUAL, ORDERED.TREE, LABEL, and LESSP, to:

```
T.
```

```
Case 1.3.1.
(IMPLIES (AND (NUMBERP Z)
              (OTP Z1)
              (OTP V1)
              (NOT (EQUAL (OT V1 Z Z1) (ET)))
              (NUMBERP D)
              (OTP C)
              (OTP X1)
              (NOT (EQUAL (OT X1 D C) (ET)))
              (NUMBERP W)
              (NOT (EQUAL (OT (OT X1 D C) W (OT V1 Z Z1))
                          (ET)))
              (NOT (EQUAL (ET)
                          (OT (OT X1 D C) W (OT V1 Z Z1))))
              (ORDERED.TREE (INSERT L (OT X1 D C)))
              (ORDERED.TREE (INSERT L (OT V1 Z Z1)))
              (NUMBERP L)
              (ORDERED.TREE (OT X1 D C))
              (ORDERED.TREE (OT V1 Z Z1))
              (LESSP D W)
              (LESSP W Z)
              (LEQ L W)
              (LESSP L W)
              (NOT (EQUAL (INSERT L (OT X1 D C)) (ET))))
         (LESSP (LABEL (INSERT L (OT X1 D C)))
                W)),
```

which further simplifies, applying RT.OT, LT.OT, and LABEL.OT, and opening up the definitions of INSERT, ORDERED.TREE, OTP, and EQUAL, to:

```
T.
```

```
Case 1.2.
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (ORDERED.TREE (INSERT L (LT X)))
              (ORDERED.TREE (INSERT L (RT X)))
              (NUMBERP L)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (LESSP (LABEL X) (LABEL (RT X)))
              (LEQ L (LABEL X))
              (LEQ (LABEL X) L))
         (ORDERED.TREE X)),
```

which again simplifies, using linear arithmetic, to:

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (ORDERED.TREE (INSERT (LABEL X) (LT X)))
              (ORDERED.TREE (INSERT (LABEL X) (RT X)))
              (NUMBERP (LABEL X))
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (LESSP (LABEL X) (LABEL (RT X)))
              (LEQ (LABEL X) (LABEL X))
              (LEQ (LABEL X) (LABEL X)))
```

```
                (ORDERED.TREE X)),
```

which we again simplify, expanding ORDERED.TREE, to:

```
      T.
```

```
Case 1.1.
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (ORDERED.TREE (INSERT L (LT X)))
                    (ORDERED.TREE (INSERT L (RT X)))
                    (NUMBERP L)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (LESSP (LABEL (LT X)) (LABEL X))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (LESSP (LABEL X) L))
               (ORDERED.TREE (OT (LT X)
                                 (LABEL X)
                                 (INSERT L (RT X))))).
```

This simplifies again, applying LABEL.OT, RT.OT, and LT.OT, and expanding the function ORDERED.TREE, to:

```
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (ORDERED.TREE (INSERT L (LT X)))
                    (ORDERED.TREE (INSERT L (RT X)))
                    (NUMBERP L)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (LESSP (LABEL (LT X)) (LABEL X))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (LESSP (LABEL X) L)
                    (NOT (EQUAL (INSERT L (RT X)) (ET))))
               (LESSP (LABEL X)
                      (LABEL (INSERT L (RT X))))).
```

Appealing to the lemma LT/LABEL/RT.ELIM, we now replace X by (OT Z W V) to eliminate (LT X), (RT X), and (LABEL X), Z by (OT X1 D C) to eliminate (LABEL Z), (RT Z), and (LT Z), V by (OT V1 Z Z1) to eliminate (LABEL V), (RT V), and (LT V), and V by (OT X1 D C) to eliminate (LABEL V), (RT V), and (LT V). We use the type restriction lemma noted when LABEL was introduced, the type restriction lemma noted when RT was introduced, and the type restriction lemma noted when LT was introduced to restrict the new variables. The result is the following five new conjectures:

```
Case 1.1.5.
        (IMPLIES (AND (EQUAL X (ET))
                      (OTP X)
                      (NOT (EQUAL (ET) X))
                      (ORDERED.TREE (INSERT L (LT X)))
                      (ORDERED.TREE (INSERT L (RT X)))
                      (NUMBERP L)
                      (ORDERED.TREE (LT X))
                      (ORDERED.TREE (RT X))
                      (LESSP (LABEL (LT X)) (LABEL X))
                      (LESSP (LABEL X) (LABEL (RT X)))
                      (LESSP (LABEL X) L)
                      (NOT (EQUAL (INSERT L (RT X)) (ET))))
                 (LESSP (LABEL X)
                        (LABEL (INSERT L (RT X))))).
```

This further simplifies, clearly, to:

T.

```
Case 1.1.4.
        (IMPLIES (AND (EQUAL V (ET))
                      (EQUAL Z (ET))
                      (OTP Z)
                      (OTP V)
                      (NUMBERP W)
                      (NOT (EQUAL (OT Z W V) (ET)))
                      (NOT (EQUAL (ET) (OT Z W V)))
                      (ORDERED.TREE (INSERT L Z))
                      (ORDERED.TREE (INSERT L V))
                      (NUMBERP L)
                      (ORDERED.TREE Z)
                      (ORDERED.TREE V)
                      (LESSP (LABEL Z) W)
                      (LESSP W (LABEL V))
                      (LESSP W L)
                      (NOT (EQUAL (INSERT L V) (ET))))
                 (LESSP W (LABEL (INSERT L V)))).
```

This further simplifies, using linear arithmetic, to:

T.

```
Case 1.1.3.
        (IMPLIES (AND (NUMBERP D)
                      (OTP C)
                      (OTP X1)
                      (NOT (EQUAL (OT X1 D C) (ET)))
                      (EQUAL Z (ET))
                      (OTP Z)
                      (NUMBERP W)
                      (NOT (EQUAL (OT Z W (OT X1 D C)) (ET)))
                      (NOT (EQUAL (ET) (OT Z W (OT X1 D C))))
                      (ORDERED.TREE (INSERT L Z))
                      (ORDERED.TREE (INSERT L (OT X1 D C)))
                      (NUMBERP L)
                      (ORDERED.TREE Z)
                      (ORDERED.TREE (OT X1 D C))
                      (LESSP (LABEL Z) W)
                      (LESSP W D)
                      (LESSP W L)
                      (NOT (EQUAL (INSERT L (OT X1 D C)) (ET))))
                 (LESSP W
                        (LABEL (INSERT L (OT X1 D C))))).
```

But this simplifies further, applying RT.OT, LT.OT, and LABEL.OT, and expanding the definitions of OTP, EQUAL, INSERT, ORDERED.TREE, LABEL, and LESSP, to:

T.

```
Case 1.1.2.
        (IMPLIES (AND (EQUAL V (ET))
                      (NUMBERP D)
                      (OTP C)
                      (OTP X1)
                      (NOT (EQUAL (OT X1 D C) (ET)))
                      (OTP V)
                      (NUMBERP W)
```

```
                    (NOT (EQUAL (OT (OT X1 D C) W V) (ET)))
                    (NOT (EQUAL (ET) (OT (OT X1 D C) W V)))
                    (ORDERED.TREE (INSERT L (OT X1 D C)))
                    (ORDERED.TREE (INSERT L V))
                    (NUMBERP L)
                    (ORDERED.TREE (OT X1 D C))
                    (ORDERED.TREE V)
                    (LESSP D W)
                    (LESSP W (LABEL V))
                    (LESSP W L)
                    (NOT (EQUAL (INSERT L V) (ET))))
               (LESSP W (LABEL (INSERT L V)))),
```

which we further simplify, applying the lemmas RT.OT, LT.OT, and LABEL.OT, and expanding the definitions of OTP, INSERT, EQUAL, ORDERED.TREE, LABEL, and LESSP, to:

```
     T.
```

```
Case 1.1.1.
     (IMPLIES (AND (NUMBERP Z)
                   (OTP Z1)
                   (OTP V1)
                   (NOT (EQUAL (OT V1 Z Z1) (ET)))
                   (NUMBERP D)
                   (OTP C)
                   (OTP X1)
                   (NOT (EQUAL (OT X1 D C) (ET)))
                   (NUMBERP W)
                   (NOT (EQUAL (OT (OT X1 D C) W (OT V1 Z Z1))
                               (ET)))
                   (NOT (EQUAL (ET)
                               (OT (OT X1 D C) W (OT V1 Z Z1))))
                   (ORDERED.TREE (INSERT L (OT X1 D C)))
                   (ORDERED.TREE (INSERT L (OT V1 Z Z1)))
                   (NUMBERP L)
                   (ORDERED.TREE (OT X1 D C))
                   (ORDERED.TREE (OT V1 Z Z1))
                   (LESSP D W)
                   (LESSP W Z)
                   (LESSP W L)
                   (NOT (EQUAL (INSERT L (OT V1 Z Z1)) (ET))))
              (LESSP W
                     (LABEL (INSERT L (OT V1 Z Z1))))),
```

which we further simplify, rewriting with RT.OT, LT.OT, and LABEL.OT, and opening up INSERT, ORDERED.TREE, OTP, and EQUAL, to:

```
     T.
```

That finishes the proof of *1. Q.E.D.

```
[1047616 cns / 1038.4 s + 144.9 gc + 27.0 io (= 1939 @ 2)]
ORDERED.TREE.INSERT
```

3.6. OCCUR.TREE.INSERT

```
Theorem.  OCCUR.TREE.INSERT:
(IMPLIES
        (AND (NUMBERP L)
             (NUMBERP K)
             (ORDERED.TREE X))
        (EQUAL (OCCUR.TREE L (INSERT K X))
               (OR (EQUAL L K) (OCCUR.TREE L X))))
```

This formula simplifies, expanding the definition of OR, to the following two new conjectures:

```
Case 2. (IMPLIES (AND (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE X)
                      (NOT (EQUAL L K)))
                 (EQUAL (OCCUR.TREE L (INSERT K X))
                        (OCCUR.TREE L X))).
```

Name the above subgoal *1.

```
Case 1. (IMPLIES (AND (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE X)
                      (EQUAL L K))
                 (EQUAL (OCCUR.TREE L (INSERT K X))
                        T)),
```

which again simplifies, trivially, to:

```
      (IMPLIES (AND (NUMBERP K) (ORDERED.TREE X))
               (OCCUR.TREE K (INSERT K X))),
```

which we would usually push and work on later by induction. But if we must use induction to prove the input conjecture, we prefer to induct on the original formulation of the problem. Thus we will disregard all that we have previously done, give the name *1 to the original input, and work on it.

So now let's consider:

```
(IMPLIES (AND (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE X))
         (EQUAL (OCCUR.TREE L (INSERT K X))
                (OR (EQUAL L K) (OCCUR.TREE L X)))),
```

named *1 above. Perhaps we can prove it by induction. Three inductions are suggested by terms in the conjecture. However, they merge into one likely candidate induction. We will induct according to the following scheme:

```
      (AND (IMPLIES (NOT (OTP X)) (p L K X))
           (IMPLIES (AND (OTP X) (EQUAL (ET) X))
                    (p L K X))
           (IMPLIES (AND (OTP X)
                         (NOT (EQUAL (ET) X))
                         (p L K (RT X))
                         (p L K (LT X)))
                    (p L K X))).
```

Linear arithmetic and the lemmas RT.LESSP and LT.LESSP inform us that the measure (COUNT X) decreases according to the well-founded relation LESSP in each induction step of the scheme. The above induction scheme generates six

new goals:

```
Case 6. (IMPLIES (AND (NOT (OTP X))
                      (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE X))
                 (EQUAL (OCCUR.TREE L (INSERT K X))
                        (OR (EQUAL L K) (OCCUR.TREE L X)))),
```

which simplifies, expanding ORDERED.TREE, to:

```
T.
```

```
Case 5. (IMPLIES (AND (OTP X)
                      (EQUAL (ET) X)
                      (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE X))
                 (EQUAL (OCCUR.TREE L (INSERT K X))
                        (OR (EQUAL L K) (OCCUR.TREE L X)))),
```

which we simplify, rewriting with RT.OT, LT.OT, and LABEL.OT, and unfolding the functions OTP, ORDERED.TREE, EQUAL, INSERT, OCCUR.TREE, and OR, to:

```
T.
```

```
Case 4. (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (ORDERED.TREE (RT X)))
                      (NOT (ORDERED.TREE (LT X)))
                      (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE X))
                 (EQUAL (OCCUR.TREE L (INSERT K X))
                        (OR (EQUAL L K) (OCCUR.TREE L X)))),
```

which simplifies, expanding the definition of ORDERED.TREE, to:

```
T.
```

```
Case 3. (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                             (OR (EQUAL L K)
                                 (OCCUR.TREE L (RT X))))
                      (NOT (ORDERED.TREE (LT X)))
                      (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE X))
                 (EQUAL (OCCUR.TREE L (INSERT K X))
                        (OR (EQUAL L K) (OCCUR.TREE L X)))).
```

This simplifies, unfolding the functions OR and ORDERED.TREE, to:

```
T.
```

```
Case 2. (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (ORDERED.TREE (RT X)))
                      (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                             (OR (EQUAL L K)
                                 (OCCUR.TREE L (LT X))))
                      (NUMBERP L)
```

```
                    (NUMBERP K)
                    (ORDERED.TREE X))
               (EQUAL (OCCUR.TREE L (INSERT K X))
                      (OR (EQUAL L K) (OCCUR.TREE L X)))).
```

This simplifies, expanding OR and ORDERED.TREE, to:

T.

```
Case 1. (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                             (OR (EQUAL L K)
                                 (OCCUR.TREE L (RT X))))
                      (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                             (OR (EQUAL L K)
                                 (OCCUR.TREE L (LT X))))
                      (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE X))
                 (EQUAL (OCCUR.TREE L (INSERT K X))
                        (OR (EQUAL L K) (OCCUR.TREE L X)))),
```

which we simplify, opening up the definitions of OR, ORDERED.TREE, INSERT, OCCUR.TREE, EQUAL, and OTP, to the following 48 new conjectures:

```
Case 1.48.
        (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (EQUAL L K))
                      (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                             (OCCUR.TREE L (RT X)))
                      (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                             (OCCUR.TREE L (LT X)))
                      (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE (LT X))
                      (ORDERED.TREE (RT X))
                      (EQUAL (LT X) (ET))
                      (EQUAL (RT X) (ET))
                      (EQUAL L (LABEL X))
                      (LEQ K (LABEL X))
                      (LESSP K (LABEL X)))
                 (EQUAL (OCCUR.TREE L
                                    (OT (INSERT K (LT X)) (LABEL X) (ET)))
                        T)).
```

However this simplifies again, rewriting with RT.OT, LT.OT, and LABEL.OT, and expanding EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

```
Case 1.47.
        (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (EQUAL L K))
                      (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                             (OCCUR.TREE L (RT X)))
                      (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                             (OCCUR.TREE L (LT X)))
                      (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE (LT X))
```

```
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (EQUAL (RT X) (ET))
                    (OCCUR.TREE L (LT X))
                    (LEQ K (LABEL X))
                    (LESSP K (LABEL X)))
               (EQUAL (OCCUR.TREE L
                                  (OT (INSERT K (LT X)) (LABEL X) (ET)))
                      T)).
```

But this simplifies again, applying LT.OT and LABEL.OT, and opening up the definitions of EQUAL, OTP, INSERT, and OCCUR.TREE, to:

```
      T.
```

Case 1.46.

```
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (NOT (EQUAL L K))
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           (OCCUR.TREE L (RT X)))
                    (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                           (OCCUR.TREE L (LT X)))
                    (NUMBERP L)
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (EQUAL (RT X) (ET))
                    (EQUAL L (LABEL X))
                    (LEQ K (LABEL X))
                    (LEQ (LABEL X) K))
               (EQUAL (OCCUR.TREE L X) T)).
```

This again simplifies, using linear arithmetic, to:

```
      T.
```

Case 1.45.

```
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (NOT (EQUAL L K))
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           (OCCUR.TREE L (RT X)))
                    (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                           (OCCUR.TREE L (LT X)))
                    (NUMBERP L)
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (EQUAL (RT X) (ET))
                    (OCCUR.TREE L (LT X))
                    (LEQ K (LABEL X))
                    (LEQ (LABEL X) K))
               (EQUAL (OCCUR.TREE L X) T)).
```

This simplifies again, using linear arithmetic, to:

```
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (NOT (EQUAL L (LABEL X)))
                    (EQUAL (OCCUR.TREE L (INSERT (LABEL X) (ET)))
```

```
                        (OCCUR.TREE L (ET)))
                  (EQUAL (OCCUR.TREE L (INSERT (LABEL X) (ET)))
                        (OCCUR.TREE L (ET)))
                  (NUMBERP L)
                  (NUMBERP (LABEL X))
                  (ORDERED.TREE (ET))
                  (ORDERED.TREE (ET))
                  (EQUAL (LT X) (ET))
                  (EQUAL (RT X) (ET))
                  (OCCUR.TREE L (ET))
                  (LEQ (LABEL X) (LABEL X))
                  (LEQ (LABEL X) (LABEL X)))
            (EQUAL (OCCUR.TREE L X) T)),
```

which again simplifies, applying LT.OT and LABEL.OT, and expanding EQUAL, OTP, INSERT, and OCCUR.TREE, to:

T.

Case 1.44.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (EQUAL (RT X) (ET))
              (EQUAL L (LABEL X))
              (LESSP (LABEL X) K))
         (EQUAL (OCCUR.TREE L
                            (OT (ET) (LABEL X) (INSERT K (RT X))))
                T)).
```

This simplifies again, applying RT.OT, LT.OT, and LABEL.OT, and expanding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

Case 1.43.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (EQUAL (RT X) (ET))
              (OCCUR.TREE L (LT X))
              (LESSP (LABEL X) K))
         (EQUAL (OCCUR.TREE L
                            (OT (ET) (LABEL X) (INSERT K (RT X))))
                T)).
```

But this again simplifies, rewriting with LT.OT and LABEL.OT, and opening up the definitions of EQUAL, OTP, INSERT, and OCCUR.TREE, to:

T.

```
Case 1.42.
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (EQUAL (RT X) (ET))
              (NOT (EQUAL L (LABEL X)))
              (NOT (OCCUR.TREE L (LT X)))
              (LEQ K (LABEL X))
              (LESSP K (LABEL X)))
         (EQUAL (OCCUR.TREE L
                            (OT (INSERT K (LT X)) (LABEL X) (ET)))
                (OCCUR.TREE L (RT X)))),
```

which again simplifies, applying RT.OT, LT.OT, and LABEL.OT, and expanding the functions EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

```
Case 1.41.
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (EQUAL (RT X) (ET))
              (NOT (EQUAL L (LABEL X)))
              (NOT (OCCUR.TREE L (LT X)))
              (LEQ K (LABEL X))
              (LEQ (LABEL X) K))
         (EQUAL (OCCUR.TREE L X)
                (OCCUR.TREE L (RT X)))),
```

which again simplifies, using linear arithmetic, to the conjecture:

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L (LABEL X)))
              (EQUAL (OCCUR.TREE L (INSERT (LABEL X) (ET)))
                     (OCCUR.TREE L (ET)))
              (EQUAL (OCCUR.TREE L (INSERT (LABEL X) (ET)))
                     (OCCUR.TREE L (ET)))
```

```
                    (NUMBERP L)
                    (NUMBERP (LABEL X))
                    (ORDERED.TREE (ET))
                    (ORDERED.TREE (ET))
                    (EQUAL (LT X) (ET))
                    (EQUAL (RT X) (ET))
                    (NOT (EQUAL L (LABEL X)))
                    (NOT (OCCUR.TREE L (ET)))
                    (LEQ (LABEL X) (LABEL X))
                    (LEQ (LABEL X) (LABEL X)))
               (EQUAL (OCCUR.TREE L X)
                      (OCCUR.TREE L (ET)))),
```

which again simplifies, applying the lemmas RT.OT, LT.OT, and LABEL.OT, and expanding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to two new conjectures:

Case 1.41.2.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NUMBERP L)
              (EQUAL (LT X) (ET))
              (EQUAL (RT X) (ET))
              (NOT (EQUAL L (LABEL X)))
              (LEQ (LABEL X) (LABEL X)))
         (NOT (OCCUR.TREE L (LT X)))).
```

But this again simplifies, expanding EQUAL, OTP, and OCCUR.TREE, to:

```
T.
```

Case 1.41.1.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NUMBERP L)
              (EQUAL (LT X) (ET))
              (EQUAL (RT X) (ET))
              (NOT (EQUAL L (LABEL X)))
              (LEQ (LABEL X) (LABEL X)))
         (NOT (OCCUR.TREE L (RT X)))),
```

which we again simplify, expanding the definitions of EQUAL, OTP, and OCCUR.TREE, to:

```
T.
```

Case 1.40.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (EQUAL (RT X) (ET))
              (NOT (EQUAL L (LABEL X)))
              (NOT (OCCUR.TREE L (LT X)))
              (LEQ (LABEL X) K))
```

```
                (EQUAL (OCCUR.TREE L
                                (OT (ET) (LABEL X) (INSERT K (RT X))))
                       (OCCUR.TREE L (RT X)))).
```

But this simplifies again, applying RT.OT, LT.OT, and LABEL.OT, and expanding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
      T.
```

```
Case 1.39.
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (NOT (EQUAL L K))
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           (OCCUR.TREE L (RT X)))
                    (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                           (OCCUR.TREE L (LT X)))
                    (NUMBERP L)
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (LESSP (LABEL (LT X)) (LABEL X))
                    (EQUAL (RT X) (ET))
                    (EQUAL L (LABEL X))
                    (LEQ K (LABEL X))
                    (LESSP K (LABEL X)))
               (EQUAL (OCCUR.TREE L
                                (OT (INSERT K (LT X)) (LABEL X) (ET)))
                      T)),
```

which we again simplify, rewriting with RT.OT, LT.OT, and LABEL.OT, and expanding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
      T.
```

```
Case 1.38.
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (NOT (EQUAL L K))
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           (OCCUR.TREE L (RT X)))
                    (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                           (OCCUR.TREE L (LT X)))
                    (NUMBERP L)
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (LESSP (LABEL (LT X)) (LABEL X))
                    (EQUAL (RT X) (ET))
                    (OCCUR.TREE L (LT X))
                    (LEQ K (LABEL X))
                    (LESSP K (LABEL X)))
               (EQUAL (OCCUR.TREE L
                                (OT (INSERT K (LT X)) (LABEL X) (ET)))
                      T)).
```

However this again simplifies, applying RT.OT, LT.OT, and LABEL.OT, and expanding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
      T.
```

```
Case 1.37.
        (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (EQUAL L K))
                      (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                             (OCCUR.TREE L (RT X)))
                      (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                             (OCCUR.TREE L (LT X)))
                      (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE (LT X))
                      (ORDERED.TREE (RT X))
                      (LESSP (LABEL (LT X)) (LABEL X))
                      (EQUAL (RT X) (ET))
                      (EQUAL L (LABEL X))
                      (LEQ K (LABEL X))
                      (LEQ (LABEL X) K))
                 (EQUAL (OCCUR.TREE L X) T)),
```

which we again simplify, using linear arithmetic, to:

```
        T.
```

```
Case 1.36.
        (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (EQUAL L K))
                      (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                             (OCCUR.TREE L (RT X)))
                      (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                             (OCCUR.TREE L (LT X)))
                      (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE (LT X))
                      (ORDERED.TREE (RT X))
                      (LESSP (LABEL (LT X)) (LABEL X))
                      (EQUAL (RT X) (ET))
                      (OCCUR.TREE L (LT X))
                      (LEQ K (LABEL X))
                      (LEQ (LABEL X) K))
                 (EQUAL (OCCUR.TREE L X) T)),
```

which again simplifies, using linear arithmetic, to:

```
        (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (EQUAL L (LABEL X)))
                      (EQUAL (OCCUR.TREE L (INSERT (LABEL X) (ET)))
                             (OCCUR.TREE L (ET)))
                      (EQUAL (OCCUR.TREE L
                                         (INSERT (LABEL X) (LT X)))
                             (OCCUR.TREE L (LT X)))
                      (NUMBERP L)
                      (NUMBERP (LABEL X))
                      (ORDERED.TREE (LT X))
                      (ORDERED.TREE (ET))
                      (LESSP (LABEL (LT X)) (LABEL X))
                      (EQUAL (RT X) (ET))
                      (OCCUR.TREE L (LT X))
                      (LEQ (LABEL X) (LABEL X))
                      (LEQ (LABEL X) (LABEL X)))
                 (EQUAL (OCCUR.TREE L X) T)),
```

which again simplifies, applying RT.OT, LT.OT, and LABEL.OT, and opening up the functions EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

```
Case 1.35.
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (EQUAL (RT X) (ET))
              (EQUAL L (LABEL X))
              (LESSP (LABEL X) K))
         (EQUAL (OCCUR.TREE L
                            (OT (LT X)
                                (LABEL X)
                                (INSERT K (RT X))))
                T)),
```

which again simplifies, rewriting with RT.OT, LT.OT, and LABEL.OT, and opening up the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

```
Case 1.34.
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (EQUAL (RT X) (ET))
              (OCCUR.TREE L (LT X))
              (LESSP (LABEL X) K))
         (EQUAL (OCCUR.TREE L
                            (OT (LT X)
                                (LABEL X)
                                (INSERT K (RT X))))
                T)).
```

This simplifies again, applying RT.OT, LT.OT, and LABEL.OT, and opening up the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

```
Case 1.33.
(IMPLIES (AND (OTP X)
```

```
                    (NOT (EQUAL (ET) X))
                    (NOT (EQUAL L K))
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           (OCCUR.TREE L (RT X)))
                    (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                           (OCCUR.TREE L (LT X)))
                    (NUMBERP L)
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (LESSP (LABEL (LT X)) (LABEL X))
                    (EQUAL (RT X) (ET))
                    (NOT (EQUAL L (LABEL X)))
                    (NOT (OCCUR.TREE L (LT X)))
                    (LEQ K (LABEL X))
                    (LESSP K (LABEL X)))
               (EQUAL (OCCUR.TREE L
                                  (OT (INSERT K (LT X)) (LABEL X) (ET)))
                      (OCCUR.TREE L (RT X)))).
```

This again simplifies, rewriting with RT.OT, LT.OT, and LABEL.OT, and expanding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

Case 1.32.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (EQUAL (RT X) (ET))
              (NOT (EQUAL L (LABEL X)))
              (NOT (OCCUR.TREE L (LT X)))
              (LEQ K (LABEL X))
              (LEQ (LABEL X) K))
         (EQUAL (OCCUR.TREE L X)
                (OCCUR.TREE L (RT X)))).
```

However this again simplifies, using linear arithmetic, to:

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L (LABEL X)))
              (EQUAL (OCCUR.TREE L (INSERT (LABEL X) (ET)))
                     (OCCUR.TREE L (ET)))
              (EQUAL (OCCUR.TREE L
                                 (INSERT (LABEL X) (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP (LABEL X))
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (ET))
              (LESSP (LABEL (LT X)) (LABEL X))
              (EQUAL (RT X) (ET))
```

```
                    (NOT (EQUAL L (LABEL X)))
                    (NOT (OCCUR.TREE L (LT X)))
                    (LEQ (LABEL X) (LABEL X))
                    (LEQ (LABEL X) (LABEL X)))
               (EQUAL (OCCUR.TREE L X)
                      (OCCUR.TREE L (ET)))),
```

which we again simplify, applying RT.OT, LT.OT, and LABEL.OT, and opening up EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (NOT (OCCUR.TREE L
                                     (INSERT (LABEL X) (LT X))))
                    (NUMBERP L)
                    (ORDERED.TREE (LT X))
                    (LESSP (LABEL (LT X)) (LABEL X))
                    (EQUAL (RT X) (ET))
                    (NOT (EQUAL L (LABEL X)))
                    (NOT (OCCUR.TREE L (LT X)))
                    (LEQ (LABEL X) (LABEL X)))
               (NOT (OCCUR.TREE L (RT X)))).
```

This again simplifies, expanding EQUAL, OTP, and OCCUR.TREE, to:

```
      T.
```

Case 1.31.

```
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (NOT (EQUAL L K))
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           (OCCUR.TREE L (RT X)))
                    (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                           (OCCUR.TREE L (LT X)))
                    (NUMBERP L)
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (LESSP (LABEL (LT X)) (LABEL X))
                    (EQUAL (RT X) (ET))
                    (NOT (EQUAL L (LABEL X)))
                    (NOT (OCCUR.TREE L (LT X)))
                    (LESSP (LABEL X) K))
               (EQUAL (OCCUR.TREE L
                                  (OT (LT X)
                                      (LABEL X)
                                      (INSERT K (RT X))))
                      (OCCUR.TREE L (RT X)))),
```

which again simplifies, rewriting with RT.OT, LT.OT, and LABEL.OT, and unfolding the functions EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
      T.
```

Case 1.30.

```
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (NOT (EQUAL L K))
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           (OCCUR.TREE L (RT X)))
                    (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
```

```
                    (OCCUR.TREE L (LT X)))
               (NUMBERP L)
               (NUMBERP K)
               (ORDERED.TREE (LT X))
               (ORDERED.TREE (RT X))
               (EQUAL (LT X) (ET))
               (LESSP (LABEL X) (LABEL (RT X)))
               (EQUAL L (LABEL X))
               (LEQ K (LABEL X))
               (LESSP K (LABEL X)))
          (EQUAL (OCCUR.TREE L
                             (OT (INSERT K (LT X))
                                 (LABEL X)
                                 (RT X)))
                 T)).
```

which we again simplify, rewriting with RT.OT, LT.OT, and LABEL.OT, and expanding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
T.
```

Case 1.29.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (LESSP (LABEL X) (LABEL (RT X)))
              (OCCUR.TREE L (LT X))
              (LEQ K (LABEL X))
              (LESSP K (LABEL X)))
         (EQUAL (OCCUR.TREE L
                            (OT (INSERT K (LT X))
                                (LABEL X)
                                (RT X)))
                T)).
```

This simplifies again, applying LT.OT and LABEL.OT, and opening up EQUAL, OTP, INSERT, and OCCUR.TREE, to:

```
T.
```

Case 1.28.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
```

```
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (EQUAL L (LABEL X))
                    (LEQ K (LABEL X))
                    (LEQ (LABEL X) K))
               (EQUAL (OCCUR.TREE L X) T)).
```

But this simplifies again, using linear arithmetic, to:

```
     T.
```

Case 1.27.

```
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (NOT (EQUAL L K))
                   (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                          (OCCUR.TREE L (RT X)))
                   (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                          (OCCUR.TREE L (LT X)))
                   (NUMBERP L)
                   (NUMBERP K)
                   (ORDERED.TREE (LT X))
                   (ORDERED.TREE (RT X))
                   (EQUAL (LT X) (ET))
                   (LESSP (LABEL X) (LABEL (RT X)))
                   (OCCUR.TREE L (LT X))
                   (LEQ K (LABEL X))
                   (LEQ (LABEL X) K))
              (EQUAL (OCCUR.TREE L X) T)),
```

which again simplifies, using linear arithmetic, to:

```
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (NOT (EQUAL L (LABEL X)))
                   (EQUAL (OCCUR.TREE L
                                      (INSERT (LABEL X) (RT X)))
                          (OCCUR.TREE L (RT X)))
                   (EQUAL (OCCUR.TREE L (INSERT (LABEL X) (ET)))
                          (OCCUR.TREE L (ET)))
                   (NUMBERP L)
                   (NUMBERP (LABEL X))
                   (ORDERED.TREE (ET))
                   (ORDERED.TREE (RT X))
                   (EQUAL (LT X) (ET))
                   (LESSP (LABEL X) (LABEL (RT X)))
                   (OCCUR.TREE L (ET))
                   (LEQ (LABEL X) (LABEL X))
                   (LEQ (LABEL X) (LABEL X)))
              (EQUAL (OCCUR.TREE L X) T)),
```

which we again simplify, applying the lemmas LT.OT and LABEL.OT, and expanding the definitions of EQUAL, OTP, INSERT, and OCCUR.TREE, to:

```
     T.
```

Case 1.26.

```
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (NOT (EQUAL L K))
                   (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                          (OCCUR.TREE L (RT X)))
                   (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                          (OCCUR.TREE L (LT X)))
```

```
                    (NUMBERP L)
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (EQUAL L (LABEL X))
                    (LESSP (LABEL X) K))
               (EQUAL (OCCUR.TREE L
                                  (OT (ET) (LABEL X) (INSERT K (RT X))))
                      T)).
```

This simplifies again, applying RT.OT, LT.OT, and LABEL.OT, and expanding EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
T.
```

```
Case 1.25.
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (NOT (EQUAL L K))
                   (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                          (OCCUR.TREE L (RT X)))
                   (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                          (OCCUR.TREE L (LT X)))
                   (NUMBERP L)
                   (NUMBERP K)
                   (ORDERED.TREE (LT X))
                   (ORDERED.TREE (RT X))
                   (EQUAL (LT X) (ET))
                   (LESSP (LABEL X) (LABEL (RT X)))
                   (OCCUR.TREE L (LT X))
                   (LESSP (LABEL X) K))
              (EQUAL (OCCUR.TREE L
                                 (OT (ET) (LABEL X) (INSERT K (RT X))))
                     T)),
```

which again simplifies, applying the lemmas LT.OT and LABEL.OT, and expanding the functions EQUAL, OTP, INSERT, and OCCUR.TREE, to:

```
T.
```

```
Case 1.24.
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (NOT (EQUAL L K))
                   (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                          (OCCUR.TREE L (RT X)))
                   (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                          (OCCUR.TREE L (LT X)))
                   (NUMBERP L)
                   (NUMBERP K)
                   (ORDERED.TREE (LT X))
                   (ORDERED.TREE (RT X))
                   (EQUAL (LT X) (ET))
                   (LESSP (LABEL X) (LABEL (RT X)))
                   (NOT (EQUAL L (LABEL X)))
                   (NOT (OCCUR.TREE L (LT X)))
                   (LEQ K (LABEL X))
                   (LESSP K (LABEL X)))
              (EQUAL (OCCUR.TREE L
                                 (OT (INSERT K (LT X))
                                     (LABEL X)
```

```
                                        (RT X)))
                              (OCCUR.TREE L (RT X)))).
```

But this again simplifies, applying RT.OT, LT.OT, and LABEL.OT, and opening up the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

Case 1.23.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (LESSP (LABEL X) (LABEL (RT X)))
              (NOT (EQUAL L (LABEL X)))
              (NOT (OCCUR.TREE L (LT X)))
              (LEQ K (LABEL X))
              (LEQ (LABEL X) K))
         (EQUAL (OCCUR.TREE L X)
                (OCCUR.TREE L (RT X)))),
```

which again simplifies, using linear arithmetic, to:

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L (LABEL X)))
              (EQUAL (OCCUR.TREE L
                                 (INSERT (LABEL X) (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT (LABEL X) (ET)))
                     (OCCUR.TREE L (ET)))
              (NUMBERP L)
              (NUMBERP (LABEL X))
              (ORDERED.TREE (ET))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (LESSP (LABEL X) (LABEL (RT X)))
              (NOT (EQUAL L (LABEL X)))
              (NOT (OCCUR.TREE L (ET)))
              (LEQ (LABEL X) (LABEL X))
              (LEQ (LABEL X) (LABEL X)))
         (EQUAL (OCCUR.TREE L X)
                (OCCUR.TREE L (RT X)))),
```

which we again simplify, applying RT.OT, LT.OT, and LABEL.OT, and unfolding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (EQUAL (OCCUR.TREE L
                                 (INSERT (LABEL X) (RT X)))
                     (OCCUR.TREE L (RT X)))
              (NUMBERP L)
```

```
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (NOT (EQUAL L (LABEL X)))
                    (LEQ (LABEL X) (LABEL X))
                    (OCCUR.TREE L (LT X)))
              (EQUAL T (OCCUR.TREE L (RT X)))),
```

which we again simplify, expanding the functions EQUAL, OTP, and OCCUR.TREE, to:

T.

```
Case 1.22.
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (NOT (EQUAL L K))
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           (OCCUR.TREE L (RT X)))
                    (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                           (OCCUR.TREE L (LT X)))
                    (NUMBERP L)
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (NOT (EQUAL L (LABEL X)))
                    (NOT (OCCUR.TREE L (LT X)))
                    (LESSP (LABEL X) K))
               (EQUAL (OCCUR.TREE L
                                  (OT (ET) (LABEL X) (INSERT K (RT X))))
                      (OCCUR.TREE L (RT X)))).
```

This simplifies again, rewriting with the lemmas RT.OT, LT.OT, and LABEL.OT, and expanding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

```
Case 1.21.
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (NOT (EQUAL L K))
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           (OCCUR.TREE L (RT X)))
                    (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                           (OCCUR.TREE L (LT X)))
                    (NUMBERP L)
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (LESSP (LABEL (LT X)) (LABEL X))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (EQUAL L (LABEL X))
                    (LEQ K (LABEL X))
                    (LESSP K (LABEL X)))
               (EQUAL (OCCUR.TREE L
                                  (OT (INSERT K (LT X))
                                      (LABEL X)
                                      (RT X)))
                      T)).
```

This simplifies again, rewriting with LABEL.OT, and expanding the definitions of OCCUR.TREE and EQUAL, to:

T.

```
Case 1.20.
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (LESSP (LABEL X) (LABEL (RT X)))
              (OCCUR.TREE L (LT X))
              (LEQ K (LABEL X))
              (LESSP K (LABEL X)))
         (EQUAL (OCCUR.TREE L
                            (OT (INSERT K (LT X))
                                (LABEL X)
                                (RT X)))
                T)),
```

which again simplifies, applying LT.OT and LABEL.OT, and opening up the functions OCCUR.TREE and EQUAL, to:

T.

```
Case 1.19.
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (LESSP (LABEL X) (LABEL (RT X)))
              (EQUAL L (LABEL X))
              (LEQ K (LABEL X))
              (LEQ (LABEL X) K))
         (EQUAL (OCCUR.TREE L X) T)).
```

This simplifies again, using linear arithmetic, to:

T.

```
Case 1.18.
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
```

```
                        (OCCUR.TREE L (LT X)))
                  (NUMBERP L)
                  (NUMBERP K)
                  (ORDERED.TREE (LT X))
                  (ORDERED.TREE (RT X))
                  (LESSP (LABEL (LT X)) (LABEL X))
                  (LESSP (LABEL X) (LABEL (RT X)))
                  (OCCUR.TREE L (LT X))
                  (LEQ K (LABEL X))
                  (LEQ (LABEL X) K))
            (EQUAL (OCCUR.TREE L X) T)),
```

which we again simplify, using linear arithmetic, to the conjecture:

```
      (IMPLIES (AND (OTP X)
                  (NOT (EQUAL (ET) X))
                  (NOT (EQUAL L (LABEL X)))
                  (EQUAL (OCCUR.TREE L
                                     (INSERT (LABEL X) (RT X)))
                        (OCCUR.TREE L (RT X)))
                  (EQUAL (OCCUR.TREE L
                                     (INSERT (LABEL X) (LT X)))
                        (OCCUR.TREE L (LT X)))
                  (NUMBERP L)
                  (NUMBERP (LABEL X))
                  (ORDERED.TREE (LT X))
                  (ORDERED.TREE (RT X))
                  (LESSP (LABEL (LT X)) (LABEL X))
                  (LESSP (LABEL X) (LABEL (RT X)))
                  (OCCUR.TREE L (LT X))
                  (LEQ (LABEL X) (LABEL X))
                  (LEQ (LABEL X) (LABEL X)))
            (EQUAL (OCCUR.TREE L X) T)).
```

This simplifies again, unfolding the functions OCCUR.TREE and EQUAL, to:

```
      T.
```

```
Case 1.17.
      (IMPLIES (AND (OTP X)
                  (NOT (EQUAL (ET) X))
                  (NOT (EQUAL L K))
                  (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                        (OCCUR.TREE L (RT X)))
                  (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                        (OCCUR.TREE L (LT X)))
                  (NUMBERP L)
                  (NUMBERP K)
                  (ORDERED.TREE (LT X))
                  (ORDERED.TREE (RT X))
                  (LESSP (LABEL (LT X)) (LABEL X))
                  (LESSP (LABEL X) (LABEL (RT X)))
                  (EQUAL L (LABEL X))
                  (LESSP (LABEL X) K))
            (EQUAL (OCCUR.TREE L
                               (OT (LT X)
                                   (LABEL X)
                                   (INSERT K (RT X))))
                   T)).
```

This simplifies again, applying LABEL.OT, and opening up the definitions of OCCUR.TREE and EQUAL, to:

```
        T.

Case 1.16.
        (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (EQUAL L K))
                      (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                             (OCCUR.TREE L (RT X)))
                      (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                             (OCCUR.TREE L (LT X)))
                      (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE (LT X))
                      (ORDERED.TREE (RT X))
                      (LESSP (LABEL (LT X)) (LABEL X))
                      (LESSP (LABEL X) (LABEL (RT X)))
                      (OCCUR.TREE L (LT X))
                      (LESSP (LABEL X) K))
                 (EQUAL (OCCUR.TREE L
                                    (OT (LT X)
                                        (LABEL X)
                                        (INSERT K (RT X))))
                        T)).
```

But this again simplifies, applying LT.OT and LABEL.OT, and expanding the functions OCCUR.TREE and EQUAL, to:

```
        T.

Case 1.15.
        (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (EQUAL L K))
                      (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                             (OCCUR.TREE L (RT X)))
                      (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                             (OCCUR.TREE L (LT X)))
                      (NUMBERP L)
                      (NUMBERP K)
                      (ORDERED.TREE (LT X))
                      (ORDERED.TREE (RT X))
                      (LESSP (LABEL (LT X)) (LABEL X))
                      (LESSP (LABEL X) (LABEL (RT X)))
                      (NOT (EQUAL L (LABEL X)))
                      (NOT (OCCUR.TREE L (LT X)))
                      (LEQ K (LABEL X))
                      (LESSP K (LABEL X)))
                 (EQUAL (OCCUR.TREE L
                                    (OT (INSERT K (LT X))
                                        (LABEL X)
                                        (RT X)))
                        (OCCUR.TREE L (RT X)))),
```

which we again simplify, rewriting with RT.OT, LT.OT, and LABEL.OT, and opening up the definition of OCCUR.TREE, to:

```
        T.

Case 1.14.
        (IMPLIES (AND (OTP X)
                      (NOT (EQUAL (ET) X))
                      (NOT (EQUAL L K))
                      (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
```

```
                    (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                    (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (LESSP (LABEL X) (LABEL (RT X)))
              (NOT (EQUAL L (LABEL X)))
              (NOT (OCCUR.TREE L (LT X)))
              (LEQ K (LABEL X))
              (LEQ (LABEL X) K))
        (EQUAL (OCCUR.TREE L X)
               (OCCUR.TREE L (RT X)))).
```

However this again simplifies, using linear arithmetic, to:

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L (LABEL X)))
              (EQUAL (OCCUR.TREE L
                                 (INSERT (LABEL X) (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L
                                 (INSERT (LABEL X) (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP (LABEL X))
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (LESSP (LABEL X) (LABEL (RT X)))
              (NOT (EQUAL L (LABEL X)))
              (NOT (OCCUR.TREE L (LT X)))
              (LEQ (LABEL X) (LABEL X))
              (LEQ (LABEL X) (LABEL X)))
         (EQUAL (OCCUR.TREE L X)
                (OCCUR.TREE L (RT X)))).
```

This simplifies again, expanding the definition of OCCUR.TREE, to:

```
T.
```

```
Case 1.13.
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (NOT (EQUAL L K))
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     (OCCUR.TREE L (RT X)))
              (EQUAL (OCCUR.TREE L (INSERT K (LT X)))
                     (OCCUR.TREE L (LT X)))
              (NUMBERP L)
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (LESSP (LABEL X) (LABEL (RT X)))
              (NOT (EQUAL L (LABEL X)))
              (NOT (OCCUR.TREE L (LT X)))
              (LESSP (LABEL X) K))
         (EQUAL (OCCUR.TREE L
                            (OT (LT X)
```

```
                                        (LABEL X)
                                        (INSERT K (RT X))))
                            (OCCUR.TREE L (RT X)))).
```

But this simplifies again, appealing to the lemmas RT.OT, LT.OT, and LABEL.OT, and unfolding the definition of OCCUR.TREE, to:

T.

Case 1.12.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (EQUAL L K)
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     T)
              (OCCUR.TREE K (INSERT K (LT X)))
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (EQUAL (RT X) (ET))
              (LEQ K (LABEL X))
              (LESSP K (LABEL X)))
         (OCCUR.TREE K
                     (OT (INSERT K (LT X))
                         (LABEL X)
                         (ET)))),
```

which we again simplify, rewriting with LABEL.OT, RT.OT, and LT.OT, and expanding the functions EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

Case 1.11.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (EQUAL L K)
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     T)
              (OCCUR.TREE K (INSERT K (LT X)))
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (EQUAL (RT X) (ET))
              (LEQ K (LABEL X))
              (LEQ (LABEL X) K))
         (OCCUR.TREE K X)),
```

which again simplifies, using linear arithmetic, to the formula:

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (EQUAL (OCCUR.TREE (LABEL X)
                                 (INSERT (LABEL X) (ET)))
                     T)
              T
              (NUMBERP (LABEL X))
              (ORDERED.TREE (ET))
              (ORDERED.TREE (ET))
              (EQUAL (LT X) (ET))
              (EQUAL (RT X) (ET))
```

```
                    (LEQ (LABEL X) (LABEL X))
                    (LEQ (LABEL X) (LABEL X)))
               (OCCUR.TREE (LABEL X) X)).
```

This again simplifies, rewriting with the lemma LABEL.OT, and expanding the functions EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
      T.
```

```
Case 1.10.
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (EQUAL L K)
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           T)
                    (OCCUR.TREE K (INSERT K (LT X)))
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (EQUAL (LT X) (ET))
                    (EQUAL (RT X) (ET))
                    (LESSP (LABEL X) K))
               (OCCUR.TREE K
                           (OT (ET)
                               (LABEL X)
                               (INSERT K (RT X))))),
```

which again simplifies, applying the lemmas LABEL.OT, RT.OT, and LT.OT, and opening up EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
      T.
```

```
Case 1.9.
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (EQUAL L K)
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           T)
                    (OCCUR.TREE K (INSERT K (LT X)))
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (LESSP (LABEL (LT X)) (LABEL X))
                    (EQUAL (RT X) (ET))
                    (LEQ K (LABEL X))
                    (LESSP K (LABEL X)))
               (OCCUR.TREE K
                           (OT (INSERT K (LT X))
                               (LABEL X)
                               (ET)))).
```

However this simplifies again, rewriting with LABEL.OT and LT.OT, and expanding the functions EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
      T.
```

```
Case 1.8.
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (EQUAL L K)
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
```

```
                    T)
               (OCCUR.TREE K (INSERT K (LT X)))
               (NUMBERP K)
               (ORDERED.TREE (LT X))
               (ORDERED.TREE (RT X))
               (LESSP (LABEL (LT X)) (LABEL X))
               (EQUAL (RT X) (ET))
               (LEQ K (LABEL X))
               (LEQ (LABEL X) K))
          (OCCUR.TREE K X)),
```

which again simplifies, using linear arithmetic, to:

```
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (EQUAL (OCCUR.TREE (LABEL X)
                                      (INSERT (LABEL X) (ET)))
                          T)
                   (OCCUR.TREE (LABEL X)
                               (INSERT (LABEL X) (LT X)))
                   (NUMBERP (LABEL X))
                   (ORDERED.TREE (LT X))
                   (ORDERED.TREE (ET))
                   (LESSP (LABEL (LT X)) (LABEL X))
                   (EQUAL (RT X) (ET))
                   (LEQ (LABEL X) (LABEL X))
                   (LEQ (LABEL X) (LABEL X)))
              (OCCUR.TREE (LABEL X) X)),
```

which we again simplify, applying LABEL.OT, and expanding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
     T.
```

Case 1.7.

```
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (EQUAL L K)
                   (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                          T)
                   (OCCUR.TREE K (INSERT K (LT X)))
                   (NUMBERP K)
                   (ORDERED.TREE (LT X))
                   (ORDERED.TREE (RT X))
                   (LESSP (LABEL (LT X)) (LABEL X))
                   (EQUAL (RT X) (ET))
                   (LESSP (LABEL X) K))
              (OCCUR.TREE K
                          (OT (LT X)
                              (LABEL X)
                              (INSERT K (RT X))))),
```

which we again simplify, rewriting with LABEL.OT, RT.OT, and LT.OT, and expanding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
     T.
```

Case 1.6.

```
     (IMPLIES (AND (OTP X)
                   (NOT (EQUAL (ET) X))
                   (EQUAL L K)
                   (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
```

```
                    T)
               (OCCUR.TREE K (INSERT K (LT X)))
               (NUMBERP K)
               (ORDERED.TREE (LT X))
               (ORDERED.TREE (RT X))
               (EQUAL (LT X) (ET))
               (LESSP (LABEL X) (LABEL (RT X)))
               (LEQ K (LABEL X))
               (LESSP K (LABEL X)))
          (OCCUR.TREE K
                      (OT (INSERT K (LT X))
                          (LABEL X)
                          (RT X)))),
```

which again simplifies, applying LABEL.OT, RT.OT, and LT.OT, and unfolding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

Case 1.5.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (EQUAL L K)
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     T)
              (OCCUR.TREE K (INSERT K (LT X)))
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (LESSP (LABEL X) (LABEL (RT X)))
              (LEQ K (LABEL X))
              (LEQ (LABEL X) K))
         (OCCUR.TREE K X)).
```

But this simplifies again, using linear arithmetic, to:

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (EQUAL (OCCUR.TREE (LABEL X)
                                 (INSERT (LABEL X) (RT X)))
                     T)
              (OCCUR.TREE (LABEL X)
                          (INSERT (LABEL X) (ET)))
              (NUMBERP (LABEL X))
              (ORDERED.TREE (ET))
              (ORDERED.TREE (RT X))
              (EQUAL (LT X) (ET))
              (LESSP (LABEL X) (LABEL (RT X)))
              (LEQ (LABEL X) (LABEL X))
              (LEQ (LABEL X) (LABEL X)))
         (OCCUR.TREE (LABEL X) X)),
```

which we again simplify, rewriting with LABEL.OT, and opening up the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

T.

Case 1.4.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (EQUAL L K)
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
```

```
                    T)
               (OCCUR.TREE K (INSERT K (LT X)))
               (NUMBERP K)
               (ORDERED.TREE (LT X))
               (ORDERED.TREE (RT X))
               (EQUAL (LT X) (ET))
               (LESSP (LABEL X) (LABEL (RT X)))
               (LESSP (LABEL X) K))
          (OCCUR.TREE K
                    (OT (ET)
                        (LABEL X)
                        (INSERT K (RT X))))).
```

This simplifies again, applying LABEL.OT, RT.OT, and LT.OT, and expanding the definitions of EQUAL, OTP, INSERT, OCCUR.TREE, and ORDERED.TREE, to:

```
T.
```

Case 1.3.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (EQUAL L K)
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     T)
              (OCCUR.TREE K (INSERT K (LT X)))
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (LESSP (LABEL X) (LABEL (RT X)))
              (LEQ K (LABEL X))
              (LESSP K (LABEL X)))
         (OCCUR.TREE K
                    (OT (INSERT K (LT X))
                        (LABEL X)
                        (RT X)))),
```

which again simplifies, rewriting with LT.OT and LABEL.OT, and expanding OCCUR.TREE, to:

```
T.
```

Case 1.2.

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (EQUAL L K)
              (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                     T)
              (OCCUR.TREE K (INSERT K (LT X)))
              (NUMBERP K)
              (ORDERED.TREE (LT X))
              (ORDERED.TREE (RT X))
              (LESSP (LABEL (LT X)) (LABEL X))
              (LESSP (LABEL X) (LABEL (RT X)))
              (LEQ K (LABEL X))
              (LEQ (LABEL X) K))
         (OCCUR.TREE K X)).
```

But this simplifies again, using linear arithmetic, to:

```
(IMPLIES (AND (OTP X)
              (NOT (EQUAL (ET) X))
              (EQUAL (OCCUR.TREE (LABEL X)
```

```
                                        (INSERT (LABEL X) (RT X)))
                                T)
                        (OCCUR.TREE (LABEL X)
                                    (INSERT (LABEL X) (LT X)))
                        (NUMBERP (LABEL X))
                        (ORDERED.TREE (LT X))
                        (ORDERED.TREE (RT X))
                        (LESSP (LABEL (LT X)) (LABEL X))
                        (LESSP (LABEL X) (LABEL (RT X)))
                        (LEQ (LABEL X) (LABEL X))
                        (LEQ (LABEL X) (LABEL X)))
                (OCCUR.TREE (LABEL X) X)).
```

This again simplifies, unfolding the definition of OCCUR.TREE, to:

```
      T.
```

```
Case 1.1.
      (IMPLIES (AND (OTP X)
                    (NOT (EQUAL (ET) X))
                    (EQUAL L K)
                    (EQUAL (OCCUR.TREE L (INSERT K (RT X)))
                           T)
                    (OCCUR.TREE K (INSERT K (LT X)))
                    (NUMBERP K)
                    (ORDERED.TREE (LT X))
                    (ORDERED.TREE (RT X))
                    (LESSP (LABEL (LT X)) (LABEL X))
                    (LESSP (LABEL X) (LABEL (RT X)))
                    (LESSP (LABEL X) K))
               (OCCUR.TREE K
                           (OT (LT X)
                               (LABEL X)
                               (INSERT K (RT X))))).
```

This again simplifies, applying RT.OT, LT.OT, and LABEL.OT, and expanding the function OCCUR.TREE, to:

```
      T.
```

That finishes the proof of *1. Q.E.D.

```
[135815 cns / 124.2 s + 17.7 gc + 43.9 io (= 735 @ 2)]
OCCUR.TREE.INSERT
```

4. Conclusion

We conclude that one should be careful when claiming that straightforward theorems cannot be proved by existing automatic theorem-provers. We encourage the authors of proof checkers and automatic theorem-provers to use their systems to prove difficult theorems, such as those reported in [3, 2, 4, 5, 6].

References

1. R. S. Boyer and J S. Moore. *A Computational Logic.* Academic Press, New York, 1979.

2. R. S. Boyer and J S. Moore. Metafunctions: Proving Them Correct and Using Them Efficiently as New Proof Procedures. In *The Correctness Problem in Computer Science*, R. S. Boyer and J S. Moore, Eds., Academic Press, London, 1981.

3. R. S. Boyer and J S. Moore. A Verification Condition Generator for FORTRAN. In *The Correctness Problem in Computer Science*, R. S. Boyer and J S. Moore, Eds., Academic Press, London, 1981.

4. R. S. Boyer and J S. Moore. MJRTY - A Fast Majority Vote Algorithm. Technical Report ICSCA-CMP-32, Institute for Computing Science and Computer Applications, University of Texas at Austin, 1982.

5. R. S. Boyer and J S. Moore. A Mechanical Proof of the Unsolvability of the Halting Problem. Technical Report ICSA-CMP-28, University of Texas at Austin, 1982.

6. R. S. Boyer and J S. Moore. Proof Checking the RSA Public Key Encryption Algorithm. Technical Report ICSCA-CMP-33, Institute for Computing Science and Computer Applications, University of Texas at Austin, 1982.

7. Agneta Eriksson, Anne-Lena Johansson, and Sten-Ake Tarnlund. Towards a Derivation Editor. Technical Report 11, Computing Science Department, Uppsala University, 1982.

DISTRIBUTION LIST

Defense Documentation Center (12 copies)
Cameron Station
Alexandria, VA 22314

Naval Research Laboratory (6 copies)
Technical Information Division
Code 2627
Washington, D.C. 20375

Office of Naval Research (2 copies)
Information Systems Program (437)
Arlington, VA 22217

Office of Naval Research
Code 200
Arlington, VA 22217

Office of Naval Research
Code 455
Arlington, VA 22217

Office of Naval Research
Code 458
Arlington, VA 22217

Office of Naval Research
Eastern/Central Regional Office
Bldg 114, Section D
666 Summer Street
Boston, MA 02210

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, IL 60605

Office of Naval Research
Western Regional Office
1030 East Green Street
Pasadena, CA 91106

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

Naval Ocean Systems Center
Advanced Software Technology Div.
Code 5200
San Diego, CA 92152

Mr. E. H. Gleissner
Naval Ship Research
& Development Center
Computation and Mathematics Dept.
Bethesda, MD 20084

Captain Grace M. Hopper (008)
Naval Data Automation Command
Washington Navy Yard
Building 166
Washington, D.C. 20374

(OCCUR.TREE L (RT X)))
(NUMBERP L)

END

FILMED

9-83

DTIC